A PAIRING ALGORITHM FOR LANDING AIRCRAFT TO CLOSELY SPACED PARALLEL RUNWAYS

Amir H. Farrahi, University of California Santa Cruz, Moffett Field, CA
Savita A. Verma, NASA Ames Research Center, Moffett Field, CA

Abstract

To facilitate pairing of aircraft while meeting a schedule, the pair-scheduling problem for landing aircraft in Very Closely Spaced Parallel Approaches was studied. An earlier prototype was adopted and the scheduling algorithm was extended in several ways to improve the solution quality and expand the range of constraints it could handle. This paper presents the scheduling problem formulation, as well as enhancements made to an earlier prototype that made it suitable for application in a human-in-the-loop simulation carried out recently at NASA Ames Research Center. Experimental data from the simulations as well as an extensive set of stress tests are analyzed and discussed. Results suggest the algorithm succeeded in suggesting aircraft pairs acceptable to the air traffic controllers in over 97% of the cases. Evaluating the performance and scalability characteristics of the algorithm demonstrates its effectiveness in discovering feasible aircraft pairs that meet all the sequencing, separation, pair-group, and runway assignment constraints. Overall, the high solution quality and short runtime makes the proposed algorithm a suitable and attractive candidate for use in a real-time aircraft-pairing application.

Introduction

Increasing the airport arrival rates is an important factor towards meeting the growing demand in air traffic. The concept of Very Closely Spaced Parallel Runway (VCSPR) operations is considered to be a crucial step for realizing significant increase in arrival throughput during poor weather conditions when instrument meteorological conditions apply. It aims to maximize the utilization in parallel runway systems that may be spaced as close as 750 ft apart, and thus increase the landing capacity at hub airports without significant increase to the airport footprint.

Several operational concepts have been developed for simultaneous approaches [1–3]. All of these concepts assume the aircraft are paired by air traffic controllers before their approach clearances are given. While there has been much research for arrival scheduling, the study by Kupfer [4] represents the only published work that specifically targeted simultaneous parallel approaches with lateral and staggered spacing of only a few seconds. Conducting research into paired arrival scheduling has other practical long-term applications as well. The insights gained from these studies could help extend the capabilities of new and improved tools for terminal area scheduling such as those currently under development [5].

This paper presents an improved algorithm for finding aircraft pairs. This algorithm was developed, and tested, and used by air traffic controllers for a HITL simulation conducted recently at NASA Ames Research Center to study the role of air traffic control [6]. The prototype development started with a Genetic Algorithm (GA) developed by Kupfer [4] as a starting point. Kupfer's algorithm was quite successful at finding pairs in the context of the simplified problem presented in [4]. However, under the requirements of the HITL simulations and based on detailed evaluation during an initial pilot study, it became apparent that Kupfer's pairing algorithm did not perform very well under the new requirements and constraints and that the algorithm required several adjustments and enhancements. Some of these enhancements were related to the problem modeling, while others targeted specific performance characteristics. This paper presents the formulation of the new pair-scheduling problem, and chronicles enhancements implemented to make it suitable for use in the HITL simulations [6]. Experimental results are presented and discussed, confirming the effectiveness of the resulting algorithm.

The rest of this paper is organized as follows. First, some introductory background is presented describing the operational concept under investigation, the motivation for developing the pairing algorithm, and an overview of the previous body of related research. Next, the formulation of the pair-scheduling problem is presented, followed by a brief discussion of the computational complexity of the pair-scheduling problem. Next, the rationale for the solution methodology adopted in the current work is provided. This is followed by the description of various modifications, adjustments and enhancements to the problem modeling, exploration and solution space control, as well as performance optimization that were implemented to improve Kupfer's algorithm in order to construct a pairing algorithm that was suitable for a real-time HITL simulation. Experimental results from the HITL simulations as well as a large number of arrival scenarios generated to conduct stress testing on the resulting pairing algorithm are presented and discussed next. The paper ends with some concluding remarks, presenting a summary of the key contributions of this work, and providing some directions for further research.

Background

The Federal Aviation Administration (FAA) recognizes that significant capacity is lost when simultaneous operations performed under visual conditions are not operational under poor weather conditions. As a part of its NextGen plan [7], the FAA aims to reduce the minimum allowable spacing between runways used for simultaneous operations in poor visibility, currently 4300 ft., by implementing revised standards and improved technologies. Table 1 shows a partial listing of airports containing parallel runways with spacing below 4300 ft. All of these airports experience severe reduction in aircraft landing capacities under poor weather conditions when simultaneous approaches are not possible.

Several concepts that address and could benefit from the revision of separation standards and technologies include Simultaneous Offset Instrument Approaches (SOIA) [8], Airborne Information Lateral System (AILS) [2] and Terminal Area Capacity Enhancing Concept (TACEC) [3]. The role of the air traffic controllers during simultaneous approaches is different for each of the above concepts. However, all of these concepts assume that the air traffic controller will assign aircraft to pairs with the knowledge that they are properly equipped. There does not exist any formal tool or process by which pairing is done, in the current NAS.

Table 1. Airports with Parallel Runways Below 4300 ft Spacing

Airport	Runway Spacing
Los Angeles (LAX)	750 ft
San Francisco (SFO)	750 ft
Seattle (SEA)	800 ft
Newark (EWR)	900 ft
Houston (IAH)	1000 ft
Las Vegas (LAS)	1000 ft
Atlanta (ATL)	1000 ft
Dallas-Ft Worth (DFW)	1200 ft
Pittsburgh (PIT)	1200 ft
St. Louis (STL)	1300 ft
Boston (BOS)	1500 ft
Orlando (MCO)	1600 ft
New York (JFK)	3000 ft
Minneapolis (MSP)	3380 ft
Memphis (MEM)	3400 ft
Raleigh (RDU)	3400 ft
Phoenix(PHX)	3565 ft
Salt Lake City (SLC)	3700 ft
Detroit (DTW)	3800 ft

A recent study conducted at NASA Ames Research Center carried out HITL simulations to investigate the allocation of tasks under TACEC for air traffic controllers [6]. This required an algorithm for pairing aircraft under different levels of automation in order to investigate the appropriate human/automation mix for the given task. The highest level of automation required a pairing algorithm that would suggest aircraft pairs to the controllers. This paper describes the problem formulation and the development of the pairing algorithm needed for these HITL simulations. In these simulations, finding and suggesting pairs was handled by automation while controllers handled the task of evaluating and accepting or rejecting the suggested pairs.

Since the pairing problem is very much related to scheduling aircraft landing, a concise overview of previous related work in scheduling is provided next. Scheduling problems arise in numerous applications and as a result, they have been studied extensively in the literature. A general survey and classification of scheduling problems can be found in [9, 10], while [11, 12] present overview of the scheduling problems encountered in aircraft and airline scheduling.

The aircraft arrival-scheduling problem has also been studied extensively. The idea of Constrained Position Shifts (CPS) was introduced in [13] as a means for improving the arrival sequence, while [14] presented a dynamic programming algorithm to optimize the aircraft arrival sequence and took advantage of CPS to increase efficiency. Other heuristics and approaches using genetic algorithms [15-19], dynamic programming [20, 21], mixed-integer linear programming with linear relaxations [22–24], branch and bound techniques [22, 35] or a combination of the above have also been applied to the arrival-scheduling problem.

The arrival scheduling problems are often broadly categorized into static [23] (offline) and dynamic (online) [17], depending on whether or not complete knowledge about the set of aircraft that are going to land is available ahead of time. In this paper, the focus is on the static case. Another classification of the arrival scheduling problems is into single vs. multiple runway landing. While much of the previous research on scheduling arrival landing focused on single runway, there have been several studies including [22, 26, 27] that discuss the multiple runway case or extend their single runway approaches to the multiple runway condition.

The scheduling problem for simultaneous landing of aircraft to very closely parallel runways is markedly different and more constrained than that of scheduling for independent landing to multiple runways. This problem was formulated recently in [4] and solved exactly using a mixed integer linear programming (MILP) formulation. As is often the case, the running time achieved using the proposed MILP formulation is prohibitively long, making the approach unsuitable for real-time application. Therefore, a heuristic technique based on genetic algorithm [28] fortified with a greedy heuristic was also presented in [4].

In this paper, the greedily enhanced genetic algorithm of Kupfer [4] was adopted as the base implementation. Then, using new requirements and feedback received during an initial pilot study, several enhancements were made to create a pairing algorithm suitable for real-time HITL simulation conducted in [6].

Concept Description

The concept investigated in the current study is TACEC [3], developed collaboratively by Raytheon and NASA Ames Research Center. The TACEC system allows multiple aircraft to fly in close formations during the final approach to enable simultaneous instrument approaches for landing aircraft on very closely spaced parallel runways that are as close as 750 ft apart. This would increase the landing capacity of the airports with closely spaced parallel runways during low visibility conditions, achieving arrival rates comparable to visual approach operations.

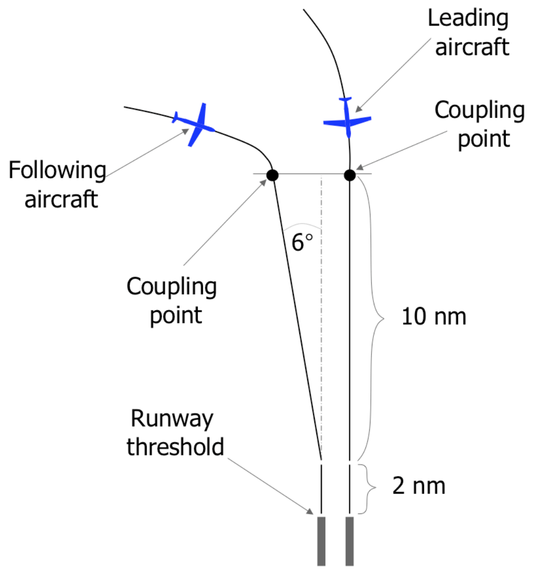

Figure 1. Approach Pattern for Aircraft Pair

The process involves pairing aircraft around 30 minutes before the aircraft arrives at the terminal boundary. As shown in Figure 1, the actual coupling for the approach is intended to occur 12 nm from the runway threshold. After this coupling point, the coupled aircraft converge over a 10nm distance at a 6° angle. For the last 2nm prior to the runway threshold, the paired aircraft would fly on parallel flight path segments.

The following aircraft in a pair must be flying within a *safe zone*, which is defined by a Lower Pairing Boundary (LPB) and an Upper Pairing Boundary (UPB) behind the lead aircraft. The LPB is defined to minimize the risk of collision in case of a blunder by the lead aircraft, while the UPB is defined in order for the follower aircraft to avoid its encounter with the wake vortex of the lead aircraft. In this study, the LPB and UPB are set to 5s and 25s behind the lead aircraft, respectively. Figure 2 demonstrates the notions of LPB and UPB and the safe zone. The pairing algorithm is instructed to schedule the pairs in order for the follower aircraft to be situated in the middle of the safe zone. The concept assumes Differential Global Positioning System (DGPS), augmented ADS-B, 4-dimensional flight management system (4D-FMS), wind detection sensors onboard the aircraft, and cockpit automation that are not extant in today's NAS.

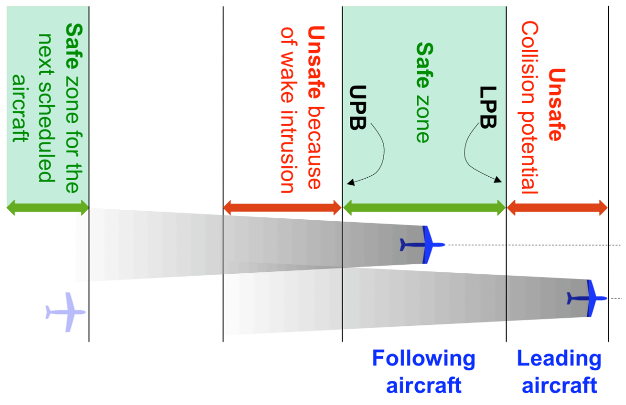
Safe zone for the next scheduled aircraft
Unsafe because of wake intrusion
UPB
Safe zone
LPB
Unsafe Collision potential
Following aircraft
Leading aircraft

Figure 2. Safe Zone for Pairing

To ensure safe operation, a minimum separation is maintained between landing aircraft that do not belong to the same pair. This separation is determined based on the wake separation category of the aircraft involved. The following four categories are recognized: Small (S), Light (L), B757 (7), and Heavy (H), in accordance with [4]. The enforced wake separation in seconds for various combinations is provided in a wake separation matrix as shown in Table 2.

Table 2. Enforced Wake Separation Matrix

		Following Aircraft			
		S	L	7	H
Leading Aircraft	S	98	83	83	72
	L	147	83	83	72
	7	180	125	125	106
	H	213	152	152	106

Another important prerequisite for the HITL simulations was a careful redesign of the airspace so that the arrival traffic can safely follow their prescribed 4D trajectories from their respective arrival streams [6]. This involved a split towards the end of the arrival streams to enable routing from each arrival stream to land an arriving aircraft on either of the two runways involved. The aerial view of the redesigned airspace used in this study, along with the geometry of the five arrival streams (Big Sur, Modesto, Oceanic, Point Reyes, Yosem) that were part of the HITL simulations are shown in Figure 3 in the vicinity of runways 28L and 28R at San Francisco airport (SFO). The flying direction of the arrival aircraft is shown using yellow arrows before the split points and using pink arrows after the split points along the arrival streams. The final portion of the paths starting from the coupling point and ending on the runways is shown in light green. This portion corresponds to the approach patterns of an aircraft pair illustrated in Figure 1.

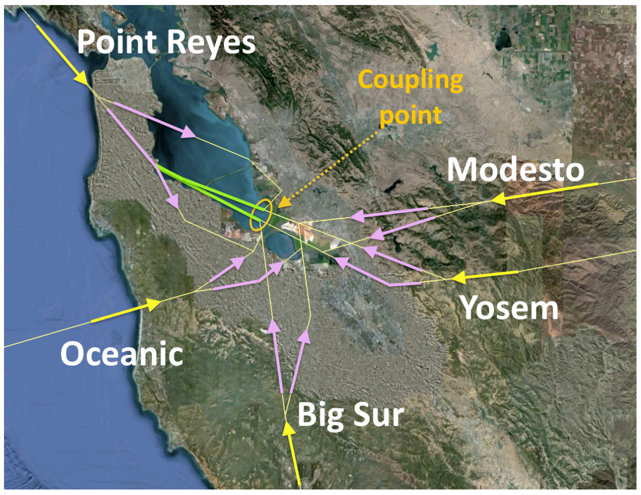

Figure 3. Arrival Stream Geometries

Formulation of the Pairing Problem

Some preliminary notations and definitions needed for problem formulation are presented first. Let $\boldsymbol{Y}=\{l, r\}$ represent the set of very closely spaced parallel (left and right) runways to land the aircraft on, and $\boldsymbol{M}=\{m_1, m_2, \ldots, m_k\}$ represent the set of arrival streams for the aircraft to be scheduled for landing. For the arrival streams used in the current study, we have $\boldsymbol{M}$={*big_sur, modesto, oceanic, point_reyes, yosem*}. These arrival streams are illustrated in Figure 3.

The set of arriving aircraft to be scheduled is denoted by $\boldsymbol{A}=\{\boldsymbol{a_1}, \boldsymbol{a_2}, \ldots, \boldsymbol{a_n}\}$, where $\boldsymbol{a_i} = (eta_{N,i}, [eta_{EL,i}, eta_{LL,i}], [eta_{ER,i}, eta_{LR,i}], w_i, s_i, g_i)$ is the record for aircraft *i*. $eta_{N,i}$ is the nominal estimated time of arrival (ETA) of aircraft *i* at the coupling point. $[eta_{EL,i}, eta_{LL,i}]$ and $[eta_{ER,i}, eta_{LR,i}]$ are the early and late ETA of aircraft *i* at the left and right coupling points, respectively. $w_i \in \{S, L, 7, H\}$, $s_i \in \boldsymbol{M}$, and g_i are the wake category, the arrival stream, and the pair-group category for aircraft *i*, respectively.

The wake separation matrix $\boldsymbol{W}$ is a *4x4* wake matrix as shown in Table 2, where entry w_{ij} is the required safe separation between non-paired aircraft of types *i* and *j*, with $i, j \in \{S, L, 7, H\}$ and *i* is ahead of *j*. The target separation window between the scheduled times of arrival (STAs) of two aircraft to be paired is denoted as t_p. For the purpose of this study, t_p= 15 seconds, and we use the wake separation matrix given in Table 2.

A scheduling solution consists of a Scheduled Time of Arrival STA_i at the coupling point and a runway assignment r_i for a (maximal) subset of *A*. A *feasible* scheduling solution is one in which a set of *constraints* are satisfied. Several classes of constraints are defined and need to be enforced. These include temporal constraints, sequencing constraints, separation constraints, pair-group constraints, and runway assignment constraints.

A *temporal constraint* places a restriction on the acceptable range for the STA values of the aircraft being scheduled for landing. In our context, the STA for an aircraft should lie within the ETA time window at the coupling point for its target runway.

A *sequencing constraint* places a restriction on the aircraft arrival sequence for the runways. In other words, it limits the acceptable order for the arrival aircraft as they pass through their respective coupling points. In our study, no overtaking is allowed among aircraft arriving from the same stream. Hence, within each arrival stream, the order of STA values for arriving aircraft to the two runways should correspond to the order of the aircraft's nominal ETA values at the coupling point.

A *separation constraint* defines the minimum safe distance among the aircraft as they fly through the airspace. In our context, the wake separation matrix and the wake categories of the arriving aircraft define the separation constraints. The separation between arriving and non-arriving aircraft is a complication that is not considered in the current study and could be a subject of future research. However, part of the requirements for redesigning the airspace and the arrival trajectories was to minimize the potential occurrence of separation conflicts between arriving and non-arriving aircraft.

Pair-group constraints impose restrictions on the set of aircraft that could be paired. This is included to enable additional level of control on which aircraft should not be paired with each other. Factors such as aircraft weight and speed profile may be used to derive the pair grouping. Two aircraft may be paired only if they belong to the same pair group.

Runway assignment constraints are the rules that define the desired or legal runway assignment for the arriving aircraft. We distinguish between single aircraft runway assignment rules, and paired aircraft runway assignment rules. The set of runway assignment rules $\boldsymbol{R}$ is the union of the set of single aircraft runway assignment rules ($\boldsymbol{R_1}$) and the set of paired aircraft runway assignment rules ($\boldsymbol{R_2}$), that is $\boldsymbol{R} = \boldsymbol{R_1} \cup \boldsymbol{R_2}$, with $\boldsymbol{R_1}=\{(m, y) \mid m \in \boldsymbol{M}, y \in \boldsymbol{Y}\}$ and $\boldsymbol{R_2}=\{((m_1, y_1), (m_2, y_2)) \mid m_1, m_2 \in \boldsymbol{M}$ and $y_1, y_2 \in \boldsymbol{Y}\}$. A single aircraft runway assignment rule $(m, y) \in \boldsymbol{R_1}$ indicates that a single aircraft arriving from stream *m* should be scheduled to arrive at runway *y*, while an aircraft-pair runway assignment rule $((m_1, y_1), (m_2, y_2)) \in \boldsymbol{R_2}$ indicates that two arrival aircraft that are going to be paired with the leader arriving from stream m_1 and the following aircraft arriving from stream m_2 should be assigned to runways y_1 and y_2 respectively. Also, it should be noted that when in conflict, a paired aircraft runway assignment rule overrides a single aircraft runway assignment rule. The runway assignment rules in use for the HITL simulations conducted in [6] will be presented later in the paper.

The Pair-Scheduling Problem (PSP) for landing single and paired aircraft to closely spaced parallel runways can be formulated as follws:

PSP *Instance*: A tuple of the form ($\boldsymbol{A}$, $\boldsymbol{W}$, t_p, $\boldsymbol{R}$), where $\boldsymbol{A}=\{a_1, a_2, \ldots, a_n\}$ is a set of aircraft records, $\boldsymbol{W}$ is a *4x4* wake separation matrix, t_p is the *pairing window* or the target separation window between two paired aircraft as they cross their respective coupling points, and $\boldsymbol{R}$ is the set of runway assignment rules.

PSP *Objective*: To find a feasible scheduling solution of a maximal subset of $\boldsymbol{A}$ subject to the temporal, sequencing, separation, pair-group, and runway assignment constraints.

To develop a pairing algorithm for solving PSP, and use it in the HITL simulations of [6], we started from the prototype provided in [4]. Then we made several enhancements that addressed some of the shortcomings based on new requirements and feedback received during an initial pilot study to make the approach suitable for use in real-time HITL simulations.

Notes on Computational Complexity

One of the key questions in studying any problem is whether or not there exists an efficient algorithm that solves the problem optimally. Answering this question in general is not an easy task. A branch of computer science commonly referred to as computational complexity is devoted to studying this matter, where problems are classified into different computational classes. As discussed earlier, the PSP problem defined earlier belongs to a general class of problems known as Scheduling problems. Many of the scheduling problems belong to the $\mathcal{NP}$–hard class of problems. Hence, finding a general solution strategy that solves an arbitrary instance of these problems optimally in a reasonable amount of time is generally considered a hopeless pursuit [9, 29].

It is not clear whether the PSP problem defined earlier belongs to the class of $\mathcal{NP}$–hard problems or not. Presenting a proof about the computational complexity of PSP requires rigorous mathematical exposition, and is beyond the scope of this paper. However, some evidence is provided that suggest PSP might belong to the $\mathcal{NP}$–hard class of problems. In particular, there exist two closely related aircraft scheduling problems that are known to be $\mathcal{NP}$–hard. These are the *aircraft landing problem*, which was studied by Beasley et al. [23], and the problem of *parallel aircraft landing with sequence-dependent separation requirements*. These problems were reported to be $\mathcal{NP}$–hard in [23] and [30], respectively.

It should be noted that during the course of this research while attempting to adopt a solution strategy, the set of requirements, constraints, and control parameters that define a feasible scheduling solution were in a constant state of flux. These requirements underwent several rounds of adjustments as various issues and limitations surfaced. Given the evidence that PSP might belong to the $\mathcal{NP}$–hard class of problems, suggested that the best course of action might be to seek algorithmic solutions that are practical, adaptable, and can produce feasible scheduling results very efficiently. This required a solution strategy that would be flexible enough to be adjusted as the requirements and constraints were being stabilized. On the other hand, we needed a prototype that could be developed in parallel and integrated into the system for conducting the real-time HITL simulations that were planned [6].

The availability of Kupfer's recent prototype [4] and its implementation based on the genetic and evolutionary algorithms [31–33] provided a very suitable candidate. The underlying genetic algorithm that was used to solve the problem could be adjusted and enhanced to suit the needs of the new problem with relative ease. These adjustments and enhancements will be explained in the following section.

Developing the Pairing Algorithm

This section describes the enhancements made to the initial basic implementation of Kupfer [4], the concerns that each enhancement was intended to address, and the significance of each of these enhancements. A brief overview of Kupfer's GA-based algorithm is presented first. This would set the stage for the later discussion where we explain the enhancements and adjustments that were made to make it suitable for real-time HITL simulations.

Kupfer's algorithm consists of three main steps. In the first step, an initial population of feasible scheduling solutions is created. The next step is the optimization phase comprised of a fixed number of iterations. During each iteration in the optimization phase, the current population of feasible solutions undergoes one generation of evolutionary optimization, followed by a greedy step to further optimize each resulting solution. In the final phase, the best solution encountered so far is reported. Of particular interest in the discussions that follow are two components of the algorithm. These are the *initialization algorithm* and the *objective function*. The initialization algorithm is responsible for seeding the genetic algorithm with its initial pool of feasible scheduling solutions. The objective function is used to determine the relative merit of a given scheduling solution and operates by associating a cost value to each candidate scheduling solution.

Problem Modeling Enhancements

The first set of enhancements made to Kupfer's greedy GA approach was applied at the problem modeling level. These adjustments intended to extend Kupfer's approach to fit the requirements of the PSP problem formulation. These enhancements included the following: *i)* implementing the Oceanic arrival stream, and *ii)* extending the algorithm to enable different ETA windows for each aircraft at the left and right coupling points.

Improved Solution Space Control

The next set of enhancements made to Kupfer's greedily enhanced GA approach was applied to enable better and more granular control over the feasible solution space. These enhancements are explained next.

Runway Assignment and Hard Pairing Rules

A major undertaking while developing the pairing algorithm was to ensure that the pairing and runway assignment rules are properly implemented and followed. These rules were iteratively defined after several rounds of discussion among the researchers and subject matter experts. The final set of pairing rules adopted for this research can be expressed as the union of single aircraft runway assignment rules $\boldsymbol{R_1}$ and the paired aircraft runway assignment rules $\boldsymbol{R_2}$ as listed below:

$$\boldsymbol{R} = \boldsymbol{R_1} \cup \boldsymbol{R_2}$$

$$\boldsymbol{R_1} = \{\ (big\_sur, l), (modesto, r), (oceanic, l), (point\_reyes, l), (yosem, r)\}$$

$$\boldsymbol{R_2} = \{\ ((point\_reyes, l), (oceanic, r)), ((point\_reyes, l), (big\_sur, r)), ((oceanic, l), (big\_sur, r)), ((yosem, l),(modesto, r))\}$$

The above rules can be explained as follows. The rules in $\boldsymbol{R_1}$ indicate that single aircrafts arriving from **Big Sur**, **Oceanic**, and **Point Reyes** stream are by default assigned to the left runway, while those arriving from **Modesto** and **Yosem** streams are by default assigned to the right runway. The rules in $\boldsymbol{R_2}$ indicate how the rules in $\boldsymbol{R_1}$ are over-ridden when aircrafts are paired. For example, the first rule ((*point_reyes*, *l*), (*oceanic*, *r*)) indicates that for an aircraft-pair arriving from the two arrival streams **Point Reyes** and **Oceanic**, the applicable runway assignment rule would be for the aircraft arriving from **Point Reyes** to land on the left runway and for the aircraft arriving from **Oceanic** to land on the right runway.

Soft Rules and Preferences

In addition to the above set of hard rules, there were some soft rules based on controller preferences. For example, it was desirable not to pair aircraft that are by default destined to the same runway, when a more or less equivalent choice existed. As an example, unless no better solution existed, it would be desirable to avoid pairing two aircraft approaching from **Modesto** and **Yosem** arrival streams. Doing so would involve a change in the runway assignment, requiring manual intervention and coordination that would lead to an increase in controller and pilot workload. Such soft rules were handled by lowering the weight assigned to a less desirable pair, in the objective function that guided the genetic algorithm during the search process.

Manual Override and Forbidden Pairs

Another important feedback received during initial pilot study of the system, was the need to allow manual override by the air traffic controllers on the pairing solution. From the beginning, the controllers had the option not to accept a suggested pair, giving them the final say in the creation of pairs. However, there was a need to communicate an undesirable pair to the pairing algorithm so that the same undesirable pair would not be suggested over and over again. This was implemented by maintaining a list of *forbidden pairs*. A forbidden pair is a pair of aircraft with designated leader and follower, which the pairing algorithm should not produce. Once the controllers identify a pair as forbidden, the pairing algorithm would add it to its list of forbidden pairs. These pairs would no longer show up as candidate pairs in the future. To implement this during the optimization, the forbidden pairs list was consulted during the creation of the initial population seed, during the course of the optimization, and once again when reporting the pairing results.

In order to gradually lead the exploration out of the undesirable portion of the search space, the objective function was used to discourage forbidden pairs from appearing in a solution by giving such pairs a much lower weight. To ensure no forbidden pairs would leak into the final pairing solution, a pruning of the pairing solution was done just before reporting the pairing result to the controllers.

Implementing and honoring the forbidden pairs feature was considered a key requirement for the HITL simulations. However, an important metric for the evaluation of the pairing algorithm is the quality of the pairing results it can produce. The goal was to ensure the pairing results produced and presented to the controllers were of high quality, so that maximum percentage of the presented pairs were accepted, thus minimizing the need for the forbidden pair feature to be exercised. In the experimental results section we will show statistics on how often the forbidden pairs features were exercised during the HITL runs.

Rewriting the Objective Function

The objective function was redesigned from scratch with all the requirements, including the new ones in mind. The revised objective function associated a cost to each scheduling solution evaluated during the course of optimization, and the genetic algorithm was setup to find the scheduling solution of minimum cost. For a given scheduling solution, the following cost function was used:

$$cost = a_0 + a_1 . P + a_2 . S + a_3 . maxSTA$$

where P is the effective number of scheduled aircraft pairs, S is the number of scheduled aircraft singles, $maxSTA$ is the largest STA among the scheduled aircraft, and a_0, a_1, a_2, a_3, are parameters used to shift and scale the cost factors appropriately. Among these, a_0 was chosen to shift the cost values to a meaningful range, and was not really significant, while a_1, a_2, a_3, were selected in order to provide the relative significance for various cost factors. The values used for these parameters in our experiments were as follows:

$$a_0 = 10{,}000$$
$$a_1 = 100$$
$$a_2 = 10$$
$$a_3 = 0.05$$

Recall from earlier discussion that certain pairs were undesirable, or forbidden. In order to allow a smooth search space exploration, an attenuation factor was used to reduce the contribution of the undesirable or forbidden pairs to P. The undesirable pairs were those involving aircraft that would require a change in their assigned runways. An attenuation factor of 0.8 was used to cut the contribution of such pairs to P by 20% as compared to a regular pair. For the forbidden pairs, an attenuation factor of 0.1 was used to make the contribution of such a pair to the objective function relatively insignificant. This would cause the algorithm to look harder for scheduling solutions that do not contain forbidden pairs. Note that we did not assign a negative cost factor to forbidden pairs, as this would cause interference with other cost factors. As a final assurance, any forbidden pairs were always filtered out of the pairing solution before reporting the list of aircraft pairs to the controllers.

Improving the Effectiveness of Finding Pairs

Another significant challenge that had to be overcome during the development phase of the pairing algorithm was to ensure that it could find and schedule as many aircraft pairs as possible. It was noticed, however, that for some relatively simple scenarios, the algorithm did not find pairs that were easily detected by simple visual inspection of the arrival streams.

Identifying Root Causes of the Problem

Careful investigation suggested that the main contributor to this degraded performance was the algorithm used for selecting the initial population pool for the genetic algorithm. In particular, two aspects of the initialization algorithm needed improvement.

The first was that the initial population was obtained by making repeated calls to a deterministic algorithm. This would result in the same exact scheduling solution to be returned upon each call, thus eliminating population diversity in the initial population pool. The lack of population diversity caused significant deterioration in the solution quality that is usually achievable within an acceptable number of generations of the genetic algorithm.

The second issue with the initial population selection was that the algorithm did not explicitly seek to find solutions containing aircraft pairs. Instead, the algorithm simply settled for any feasible solution regardless of whether or not it contained any aircraft pairs. These two issues combined, often resulted in an initial pool of individual solutions consisting of multiple copies of the same exact feasible solution not containing any aircraft pairs.

As a result, the initial population was often quite suboptimal and contained no diversity. This resulted in a significant impediment to the genetic algorithm's ability to escape from the local minima and find solutions of superior quality.

New GA Initialization Algorithm

To address the quality degradation issue and to boost the algorithm's ability in finding adequate number of aircraft pairs, a new and improved GA initialization algorithm was developed. The new approach still used the idea of calling the same (new) algorithm multiple times to produce the seed population for the genetic algorithm. However, special attention was given to two aspects of the algorithm, based on the root causes identified for degraded quality, as discussed earlier. First, the new algorithm was non-deterministic, producing different feasible solutions on consecutive calls. Secondly, the algorithm was explicitly designed to target finding feasible solutions that maximize the number of aircraft pairs.

This was done using a very efficient greedy non-deterministic algorithm for solving the PSP problem that worked off of a partial schedule, which was constructively formed starting from an empty set. The key observation was that at any given moment in time, the set of choices for the next aircraft to be scheduled for landing is the set of aircraft leading the pack in each of the arrival streams. In other words, the algorithm would only need to focus on the first unscheduled aircraft on each arrival stream, arriving the coupling point. This observation was utilized to reduce the search space significantly. This was achieved by maintaining the set of arrival aircraft approaching on different streams using separate queues, and focusing on the first candidate in each queue when searching for the next aircraft to be scheduled.

The non-determinism was achieved by selecting a random STA for the best candidate within the feasible range, while satisfying all the constraints. Since the focus was still among the first arriving aircraft from each arrival stream, this algorithm is a deterministic greedy pairing algorithm using the First Come First Served (FCFS) heuristic. The outline of the algorithm is given below:

1. Create an empty Schedule.
2. Create arrival stream queues, each sorted in the increasing order of the nominal ETA values.
3. Pop the first aircraft from each non-empty queue and add it to candSet .
4. If candSet is empty no more aircraft can be scheduled legally. Report the schedule constructed so far and exit.
5. Otherwise, evaluate each of the aircraft in candSet to see if it can be legally scheduled, given the partial schedule constructed so far. Remove from candSet any aircraft that cannot be scheduled legally.
6. Let $B \in$ candSet be the aircraft that could be scheduled earliest while satisfying all constraints, or $B = \emptyset$ if candSet is empty.
7. If ($B = \emptyset$), go to step 3.
8. Remove B from candSet, schedule B randomly in its legal range.
9. Pop the next aircraft from the stream containing B (if there exists one) and add it to candSet.
10. Go to step 4

Much of the algorithm is quite straightforward. The most interesting and involved steps in this algorithm are steps 5 and 6. Step 5 is a pruning step to eliminate from consideration any aircraft in the candSet that cannot be legally scheduled given the partial schedule that is constructed so far. Note that if an aircraft cannot be legally scheduled, given the partial schedule that is constructed so far, there is no need to consider it in the future either. This is because of the constructive and greedy nature of the algorithm. Therefore, we can safely eliminate these aircraft from consideration. Step 6 seeks to find the best among the remaining candidate aircraft in candSet. It is implemented using a case-based analysis that looks at whether or not the last scheduled aircraft was single or paired. Then it looks at all the choices to pick the aircraft that can be scheduled as early as possible, while satisfying all the scheduling constraints. The choice that gives the minimum possible STA value meeting all the wake separation, pairing, pair grouping, and other constraints is returned as the best choice stored in B.

Once the best choice among the candidate aircraft is identified, it is scheduled within its legal range in a randomized fashion in step 8. This step provides the desired non-determinism in the initial population pool, resulting in diversity of the seed population. The genetic algorithm can then use this diversity for much more effective exploration of the solution space using genetic mutation and crossover operations.

Experimental Results

The pairing algorithm described in this paper was implemented in C/C++ using the Genetic Algorithm Library GAlib [34] and integrated into the ground air traffic control system for HITL simulations [6]. A stand-alone version was also implemented to conduct off-line performance and stress testing. The stand-alone runtimes are performed on a Mac Pro machine with 2 x 2.8 GHz Quad-Core Intel Xeon processor. This section presents the results of experiments conducted to measure the performance, scalability, and quality of the pairing results produced by the pairing algorithm.

Performance Results

The first set of results, presented in Table 3, shows the impact of the enhancements made to the GA initialization algorithm on the number of pairs discovered by the pairing algorithm. Each row of the table presents one of the scenarios that were tested. The first six test cases (row 1 through 6) were chosen randomly from those that were being developed in preparation of the HITL simulations [6]. The last test case (row 7) shows the scenario used to help identify the issues with the original GA initialization algorithm.

As Table 3 shows, with the new initialization algorithm, significantly more number of aircraft pairs were found and scheduled. In many cases the original algorithm failed to find any pairs, even after evolving for several thousand generations. The total numbers of discovered pairs in these runs were 6 and 40, using the original and improved GA initialization algorithms, respectively. Note that this improvement was achieved at the expense of only about 5% increase in the overall runtime.

Table 3. Impact of GA Initialization Algorithm

		Original		**Improved**	
Test	**# A/C**	**#pairs**	**runtime (s)**	**#pairs**	**runtime (s)**
1	11	0	1	**4**	**2**
2	12	1	3	**5**	**3**
3	14	5	4	**7**	**4**
4	16	0	2	**6**	**3**
5	18	0	2	**5**	**2**
6	20	0	3	**6**	**3**
7	34	0	6	**7**	**5**
Total		**6**	**21**	**40**	**22**

The scenarios developed for the HITL simulations [6] were intended to exercise the system in different operating modes. The main objective in preparing these scenarios was to provide coverage for the operational procedures that provide insights into the proper allocation of tasks under TACEC for automation and the air traffic controllers. These scenarios were quite helpful in revealing certain limitations of the original implementation of the pairing algorithm and identifying potential areas that needed enhancements. However, they were not developed to stress the pairing algorithm sufficiently. In order to conduct a more detailed evaluation of the new GA initialization algorithm, a large suite of stress test scenarios were generated using a randomized parametric algorithm for generating arrival scenarios. The parameters included the number of arrival streams, the number of aircraft, the number of pair groups, and the minimum and maximum spacing requirements between the arrival times of consecutive aircraft on the same arrival stream.

Experiments were conducted to see the impact of the GA initialization algorithm on the performance of the pairing algorithm. Table 4 shows the results of these experiments. Each row in this table corresponds to over two hundred different scenarios, each with the same number of aircraft. The average number of pairs and the average runtime in seconds are shown on each row for all the runs with the same number of aircraft in the scenario.

The table shows the average value for the number of pairs and runtime across all the runs with the same number of aircraft in the scenario as specified in the first column. It can be seen that the pairing algorithm with the original GA initialization algorithm had significant difficulty in scheduling aircraft pairs, while the implementation of the pairing algorithm using the improved version of the GA initialization algorithm found many pairs. This came at the expense of about three-fold increase in runtime, which is reasonable considering the improved performance in terms of the number of aircraft pairs scheduled. Besides, as discussed in the next section the runtimes remained quite manageable and within the expected response time requirements for a real-time application.

Table 4. Stress Test Scenario Result Averages

	Original		Improved	
# A/C	**#pairs**	**runtime (s)**	**#pairs**	**runtime (s)**
10	0.03	1.20	**1.42**	**2.11**
20	0.02	2.16	**4.30**	**4.64**
30	0.00	3.17	**7.12**	**7.40**
40	0.00	4.26	**11.76**	**10.66**
50	0.00	5.47	**15.76**	**14.26**
60	0.00	6.64	**18.75**	**18,28**
70	0.00	7.83	**21.62**	**22.02**
80	0.00	9.04	**25.02**	**26.70**
90	0.00	10.31	**29.93**	**30.84**
100	0.00	11.50	**32.74**	**35.54**
Total	**0.05**	**61.58**	**168.42**	**173.45**

To show the total number of aircraft that could be landed in each scenario, we need to include the number of single aircraft that were scheduled for landing as well. Figure 4 shows the total number of aircraft scheduled for landing on average, for the stress test scenarios that were used to obtain the results presented in Table 4.

As shown in Figure 4, the implementation with the original GA initialization algorithm was not very successful at finding and scheduling pairs of aircraft for simultaneous landing. As a result, the total number of aircraft that were scheduled for landing was considerably lower when we used the original GA initialization algorithm, although the number of aircraft scheduled for single landing was generally equal or more than those obtained using the improved GA initialization algorithm.

In the rest of the experimental results section, unless explicitly mentioned otherwise, the improved GA initialization algorithm is used and implied for all the reported results.

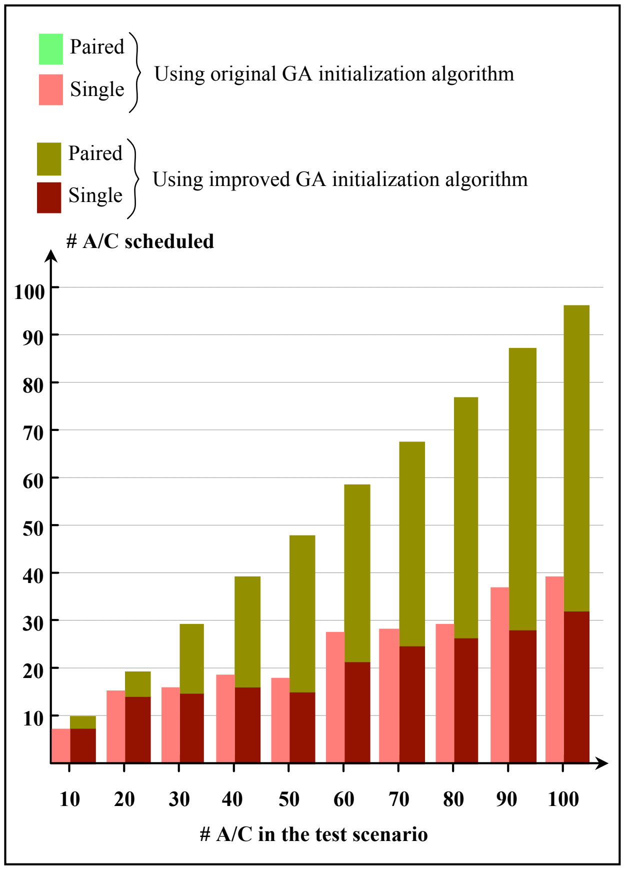

Figure 4. Impact of the Initialization Algorithm

Efficiency and Scalability Results

An important characteristic of a pairing algorithm to be used for real-time applications is a measure of its scalability. An algorithm is considered efficient if its runtime scales sufficiently slowly. This allows its application on large instances and results in response times that are acceptable for real-time applications. The same stress test scenarios used to generate the results for Table 4 were employed to test the scalability of the GA based pairing algorithm.

Figures 5 and 6 show the percentage of aircraft successfully scheduled for landing, and the runtime of the pairing algorithm, respectively, as a function of the number of aircraft in the scenario. In Figure 5, the percentage of aircraft successfully scheduled is further broken down into its paired-landing and single-landing constituents.

As shown in Figure 5, the percentage of the total aircraft successfully scheduled for landing stays at a respectable value, well above 95% even for problem instances containing up to one hundred aircraft. At the same time, the percentage of the aircraft that are scheduled for paired landing comprises a significant portion of the total, ranging from 30% to more than 60% of all the aircraft. This means that the algorithm performs in a very scalable fashion with no sign of deterioration as the number of arriving aircraft in the problem instance approaches one hundred or more.

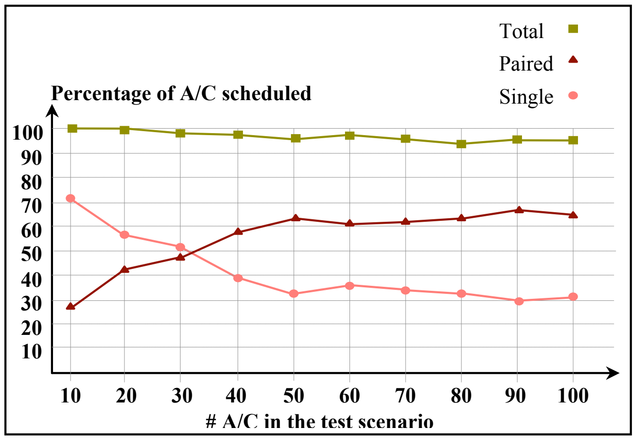

Figure 5. Percentage of Scheduled Aircraft

As Figure 6 shows, a 100-aircraft instance of the problem required no more than 40 seconds to solve, on average. Most of the scenarios in the HITL simulations contained 60 or fewer aircraft, requiring no more than 20 seconds to solve, on average. As discussed earlier, the pairing occurs about 30 minutes prior to the aircraft reach the terminal area boundary. A pairing algorithm with less than half a minute of turnaround time proved quite adequate in the HITL simulation that was conducted. On most instances the pairing result was available in less than 10 seconds since the number of aircraft in the problem instance was often 30 or less.

Quality of the Scheduling Results

Note that ultimately making a judgment call about the quality and suitability of an aircraft pair identified for simultaneous landing is the responsibility of the human operators and air traffic controllers. To measure how good the pairing algorithm's results were, the HITL simulations were examined to see how often the controllers exercised the forbidden pair feature. The result of this investigation is shown in Figure 7.

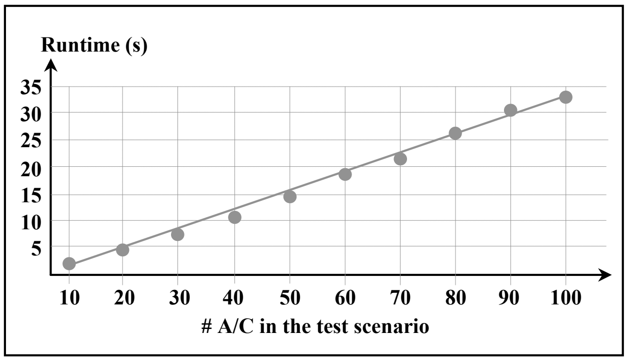

Figure 6. Runtime Profile of the Algorithm

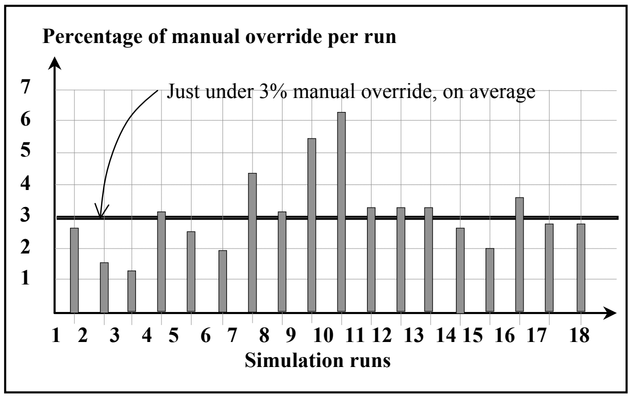

Figure 7. Percentage of Manual Override Usage

The data that was collected during the HITL simulation consisted of 18 runs, each representing a few hours of simulation. During the course of these experiments the pairing scheduler suggested a total of 9,381 pairs. The air traffic controllers accepted all pairs except for 281 (less than 3%), which were identified as forbidden pairs. In other words, the pairing algorithm produced results that were acceptable in over 97% of the instances. This is a very encouraging result and speaks to the effectiveness of the objective function that was developed to help guide the search process.

Conclusion

An improved pairing algorithm for Very Closely Spaced Parallel Approaches is presented. An earlier greedily enhanced genetic algorithm prototype was adopted, extended and enhanced in several ways to make it suitable for automatic pairing and scheduling of paired and single aircraft and for use in a recent HITL simulation. The key contributions of this paper include:

- Extending the problem formulation and extending the basic Implementation to allow its application to the extended problem.
- Evaluating the performance of the basic algorithm and identifying GA initialization algorithm as root cause of its degraded performance.
- Implementation of a much more effective randomized GA initialization algorithm that specifically seeks to maximize the number of pairs while meeting all the constraints.
- Development of a simplified and improved objective function to guide the search process.
- Putting in place proper controls in the algorithm to guarantee strict adherence to hard rules and preferential adherence to soft rules.
- Implementation of forbidden-pair feature to enable air traffic controllers to instruct the pairing algorithm about undesirable pairs.

The resulting algorithm was integrated and used successfully in recent HITL simulations conducted at NASA Ames Research Center. In addition to the scheduling scenarios that were part of the HITL simulations, an extensive set of stress test scenarios were developed to exercise the pairing algorithm more thoroughly. Experimental results indicate:

- The runtime of the proposed algorithm scales linearly with the number of aircraft in the problem instance, with acceptable response time making it suitable for real-time application.
- The proposed pairing algorithm succeeded in finding significantly more number of pairs, compared to its predecessor.
- The pairing results quality was high despite its fast running time. Analysis of the HITL simulation results indicated that over 97% of the pairs suggested by the algorithm were accepted by the air traffic controllers.

Some generalizations of the problem that might be of interest for future research include extensions to more than two parallel runways, and inclusion of additional constraints and objectives such as balancing the traffic on the runways, as well as simultaneous consideration of arrival and departure traffic. Conducting research into paired arrival and departure scheduling has other practical long-term applications. The insights gained from these studies could help extend the capabilities of new and improved tools for terminal area scheduling such as those currently under development [5].

References

[1] Bone, R., A. Mundra, B.O. Olmos, 2001, “Paired Approach Operational Concepts,” Digital Avionics Systems Conference, Daytona Beach, FL.

[2] Abbott, T. & Elliot, D., 2001, “Simulator Evaluation of Airborne Information for lateral Spacing (AILS) Concept,” NASA/TP-2001-210665.

[3] Miller, M.E., S. Dougherty, J. Stella, P. Reddy, 2005, “CNS Requirements for Precision Flight in Advanced Terminal Airspace,” IEEE Aerospace Conference, Big Sky, MO.

[4] Kupfer, M., 2009, “Scheduling Aircraft Landings to Closely Spaced Parallel Runways,” Eighth USA/Europe Air Traffic Management Research and Development Seminar, Napa, CA.

[5] Thipphavong, J., D. Mulfinger, A. Sadovsky, 2010, “Design Considerations for a New Terminal Area Scheduler”, AIAA Aviation Technology, Integration, and Operations, Fort Worth, TX.

[6] Verma, S., T. Kozon, Deborah Ballinger, 2010, “Preliminary Guidelines – Air Traffic Control

Procedures for Pairing Aircraft for Closely Spaces Simultaneous Approaches," Applied Human Factors Ergonomics Conference, Miami, FL.

[7] Federal Aviation Administration, 2008, " National Aviation Research Plan," http://nas-architecture.faa.gov/nas/downloads/

[8] Federal Aviation Administration, 2006, Order 8260.49A, "Simultaneous Offset Instrument Approach (SOIA)."

[9] Brucker, P., 2009, *Scheduling Algorithms*, Fifth Edition, Springer-Verlag, Berlin, Germany.

[10] Graham, R.L., E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, 1979, "Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey," Annals of Discrete Mathematics, Vol. 5, pp. 278–326.

[11] Gopalan, R., K.T. Talluri, 1998, "Mathematical Models in Airline Schedule Planning: A Survey," Annals of Oper. Res., Vol. 76, pp. 155–185.

[12] Etschmaier, M.M., D.F.X. Mathaisel, 1984, "Aircraft Scheduling – The State of the Art," 24th AGIFORS Annual Symposium, pp. 181– 225.

[13] Dear, R.G., 1976 "The Dynamic Scheduling of Aircraft in the Near Terminal Area," Research Report R76-9, MIT Flight Transportation Laboratory, Cambridge, MA.

[14] Psaraftis, H.N., 1978 "A Dynamic Programming Approach to the Aircraft Sequencing Problem," Research Report R78–4, MIT Flight Transportation Laboratory, Cambridge, MA.

[15] Ciesielski, V., P. Scerri, 1997, "An Anytime Algorithm for Scheduling Aircraft Landing Times Using Genetic Algorithms," Australian J. of Intelligent Information Processing Systems, Vol. 4, pp. 206–213.

[16] Grosche, T., A. Heinzl, F. Rothlauf, 2001, "A Conceptual Approach for Simultaneous Flight Schedule Construction with Genetic Algorithms," Applications of Evolutionary Computing, Lecture Notes in Computer Science, Vol. 2037/2001, pp. 257–267, Springer.

[17] Beasley, J.E., M. Krishnmoorthy, Y.M. Sharaiha, D. Abramson, 2004, "Displacement Problem and Dynamically Scheduling Aircraft Landings," J. of the Operational Research Society, Vol. 55, Issue 1, pp. 54–64, Palgrave, Macmillan.

[18] Hu, X.-B., W.-H. Chen, 2005, "Genetic Algorithm based on Receding Horizon Control for Aircraft Sequencing and Scheduling," J. of Engineering Approaches of Artificial Intelligence, Vol. 18, No. 5, pp. 633–642.

[19] Hu, X.-B., E. Di Paolo, 2009, "An Efficient Genetic Algorithm with Uniform Crossover for Air Traffic Control," Computers and Operations Research, Vol. 36, Issue 1, pp. 245–259.

[20] Chandran, B., H. Balakrishnan, 2007, " Dynamic Programming Algorithm for Robust Runway Scheduling, " American Control Conference, New York, NY.

[21] Lee, H., H. Balakrishnan, 2008, "A Study of Tradeoffs in Scheduling Terminal Area Operations," Proceedings of the IEEE, Vol. 96, No. 12, pp. 2081–2095.

[22] Ernst, A.T., M. Krishnamoorty, R.H. Storer, 1999, "Heuristic and Exact Algorithms for Scheduling Aircraft Landings," Networks Int'l J., Vol. 34, No. 3, pp. 229–241.

[23] Beasley, J.E., M. Krishnamoorthy, Y.M. Sharaiha, D. Abramson, 2000 "Scheduling Aircraft Landings – The Static Case," Transportation Science, Vol. 34, Issue 2, pp. 180–197.

[24] Roy, K., A.M. Bayen, C.J. Tomlin,2005, "Polynomial Time Algorithms for Scheduling of Arrival Aircraft," AIAA Guidance, Navigation anc Control, San Francisco, CA.

[25] Abela, J., D. Abdamson, M. Krishnamoorthy, A. De Silva, G. Mills, 1993, "Computing Optimal Scheduling for Landing Aircraft," Proc. 12th National ASOR Conference, Adelaide, Australia, pp. 71–90.

[26] Balakrishnan, H., B. Chandran, 2006, "Scheduling Aircraft Landings under Constrained Position Shifting," AIAA Guidance, Navigation, and Control Conference and Exhibit, Keystone, CO.

[27] Jani, M., 2008, "Modeling the Capacity of Closely-Spaced Parallel Runways using Innovative Approach Procedures," Transportation Res., Part C: Emerging Technologies, Vol. 16(6), pp. 704–730.

[28] Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional.

[29] Garey, M.R., D.S. Johnson, 1979, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman and Co., New York.

[30] Bayen, A.M., T. Callantine, C.J. Tomlin, Y. Ye, J. Zhang, 2004, "Optimal Arrival Traffic Spacing via Dynamic Programming," AIAA Conference on Guidance, Navigation and Control, Providence, RI

[31] Goldberg, D.E., 1989, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley Professional.

[32] Eiben, A.E., J.E. Smith, 2003, *Introduction to Evolutionary Computing*, Springer.

[33] Haupt, R.L., S.E Haupt, 2004, *Practical Genetic Algorithms*, Second Edition, Wiley Interscience.

[43] Wall, M., 1996, "GAlib: A C++ Library of Genetic Algorithm Components," Mechanical Engineering Departement, Massachusetts Institute of Technology, http://lancet.mit.edu/ga/. Save.

29th Digital Avionics Systems Conference
October 3-7, 2010